

# The Playlist Store

---

Understanding how the store is implemented is beyond the scope of this course

Databases, how they are designed and implemented is the subject of later courses

We can regard the store as a 'black box' - which can store and retrieve objects

All object have ids.

Objects can contain other objects

#### back-end +

```
⌘ .env  
.gitignore  
.jscsrc  
controllers/about.js  
controllers/dashboard.js  
controllers/playlist.js  
controllers/start.js  
models/json-store.js  
models/playlist-store.js  
models/playlist-store.json  
package.json  
README.md  
routes.js  
server.js  
utils/logger.js
```

#### front-end +

```
📁 assets  
views/about.hbs  
views/dashboard.hbs  
views/layouts/main.hbs  
views/partials/addplaylist.hbs  
views/partials/addsong.hbs  
views/partials/listplaylists.hbs  
views/partials/listsongs.hbs  
views/partials/menu.hbs  
views/playlist.hbs  
views/start.hbs
```

*models* contains the database module

- **json-store.js**: a module to manage a database
- **playlist-store.js**: a module to manage a database of playlist objects
- **playlist-store.json**: the actual database itself - a simple file

# playlist-store.json

The actual database file itself.

The database module read/write this file.

In Gomix - the contents of the file may not be immediately in sync with the app

Restarting the app may be need to keep the editor in sync

```
{
  "playlistCollection": [
    {
      "id" : "01",
      "title": "Beethoven Sonatas",
      "duration": 35,
      "songs": [
        {
          "id" : "04",
          "title": "Piano Sonata No. 3",
          "artist": "Beethoven",
          "genre": "Classical",
          "duration": 5
        },
        {
          "id" : "05",
          "title": "Piano Sonata No. 7",
          "artist": "Beethoven",
          "genre": "Classical",
          "duration": 6
        },
        {
          "id" : "06",
          "title": "Piano Sonata No. 10",
          "artist": "Beethoven",
          "genre": "Classical",
          "duration": 4
        }
      ]
    },
    {
      "id" : "02",
      "title": "Beethoven Concertos",
      "duration": 23,
      "songs": [
        {
          "id" : "07",
          "title": "Piano Concerto No. 0",
          "artist": "Beethoven",
          "genre": "Classical",
          "duration": 8
        },
        {
          "id" : "08",
          "title": "Piano Concerto No. 4",
          "artist": "Beethoven",
          "genre": "Classical",
          "duration": 3
        },
        {
          "id" : "09",
          "title": "Piano Concerto No. 6",
          "artist": "Beethoven",
          "genre": "Classical",
          "duration": 4
        }
      ]
    }
  ]
}
```

# json-store.js

General purpose  
module to maintain  
a JSON database

No need to modify  
this module

Uses the **lowdb**  
database module

<https://github.com/typicode/lowdb>

```
'use strict';

const low = require('lowdb');
const fileAsync = require('lowdb/lib/file-async');

class JsonStore {
  constructor(file, defaults) {
    this.db = low(file, { storage: fileAsync, });
    this.db.defaults(defaults).value();
  }

  add(collection, obj) {
    this.db.get(collection).push(obj).last().value();
  }

  remove(collection, obj) {
    this.db.get(collection).remove(obj).value();
  }

  removeAll(collection) {
    this.db.get(collection).remove().value();
  }

  findAll(collection) {
    return this.db.get(collection).value();
  }

  findOneBy(collection, filter) {
    const results = this.db.get(collection).filter(filter).value();
    return results[0];
  }

  findByIds(collection, ids) {
    return this.db.get(collection).keyBy('id').at(ids).value();
  }

  findBy(collection, filter) {
    return this.db.get(collection).filter(filter).value();
  }
}

module.exports = JsonStore;
```

# playlist-store.js

Module to manage  
playlists

No need to modify  
this module

Uses the JsonStore  
module to manage  
the database

```
'use strict';

const _ = require('lodash');
const JsonStore = require('./json-store');

const playlistStore = {

  store: new JsonStore('./models/playlist-store.json', { playlistCollection: 'playlistCollection' },

  getAllPlaylists() {
    return this.store.findAll(this.collection);
  },

  getPlaylist(id) {
    return this.store.findOneBy(this.collection, { id: id });
  },

  addPlaylist(playlist) {
    this.store.add(this.collection, playlist);
  },

  removePlaylist(id) {
    const playlist = this.getPlaylist(id);
    this.store.remove(this.collection, playlist);
  },

  removeAllPlaylists() {
    this.store.removeAll(this.collection);
  },

  addSong(id, song) {
    const playlist = this.getPlaylist(id);
    playlist.songs.push(song);
  },

  removeSong(id, songId) {
    const playlist = this.getPlaylist(id);
    const songs = playlist.songs;
    _.remove(songs, { id: songId });
  },
};

module.exports = playlistStore;
```

# playlist-store.js

Use this API to  
manage the store

Contains all the  
methods the app  
needs

Controllers don't  
need to know how  
these methods are  
implemented - they  
just assume they  
work

```
const playlistStore = {  
  getAllPlaylists() {  
    ...  
  },  
  getPlaylist(id) {  
    ...  
  },  
  addPlaylist(playlist) {  
    ...  
  },  
  removePlaylist(id) {  
    ...  
  },  
  removeAllPlaylists() {  
    ...  
  },  
  addSong(id, song) {  
    ...  
  },  
  removeSong(id, songId) {  
    ...  
  },  
};
```